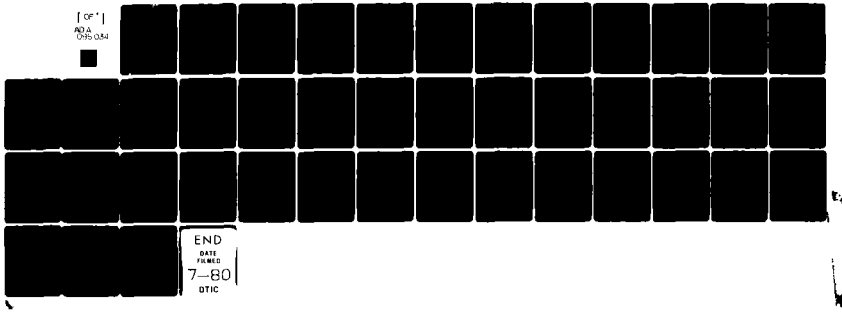
AD-A085 034 ROYAL AIRCRAFT ESTABLISHMENT FARNBOROUGH (ENGLAND) F/G 9/2
A FREE-FORMAT DATA INPUT SCHEME WRITTEN IN FORTRAN IV,(U)
MAY 79 G F BUTLER, J PIKE
UNCLASSIFIED RAE-TM-AERO-1804 DRIC-BR-69582 NL



TECH. MEMO
AERO 1804

TECH. MEMO
AERO 1804

ROYAL AIRCRAFT ESTABLISHMENT

A FREE-FORMAT DATA INPUT SCHEME WRITTEN IN FORTRAN IV

by

G. F. Butler
J. Pike

May 1979

ROYAL AIRCRAFT ESTABLISHMENT

Technical Memorandum Aero 1804

Received for printing 14 May 1979

A FREE-FORMAT DATA INPUT SCHEME WRITTEN IN FORTRAN IV

by

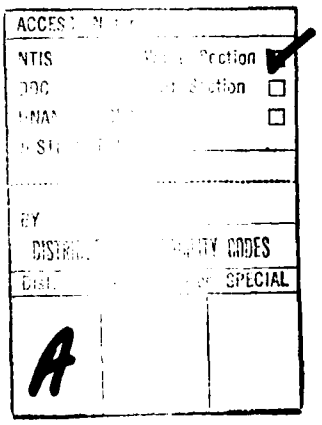
G.F. Butler

J. Pike

SUMMARY

Data for Fortran IV programs has to be in fixed format. Some users find this constraint irksome. Presented here is a data input scheme for Fortran IV which makes the minimum of demands on the data structure. If the data is well-formed and unambiguous it will be read. Ill-formed and ambiguous data will also be read and given a reasonable interpretation, with the location of the suspect data and the value assumed being output as a (suppressible) error message. The scheme also allows input variables to retain their previous values, identical consecutive data to be input in a simplified form and alphanumeric comments to appear amongst the data. The whole data input scheme is written in standard Fortran IV so that the advantages of machine transferability of the program are retained.

LIST OF CONTENTS



1 INTRODUCTION

Programs written in standard Fortran IV are limited to data input with fixed format. Often it is more convenient to input data separated by commas or spaces, for example, and in recognition of this various manufacturers have introduced their own 'free-format' data input routines. Unfortunately these are not standard and programs and data prepared for input in free-format on one manufacturer's machine cannot, in general, be transferred to those of another manufacturer. In this Memorandum free-format data input routines, *written in standard Fortran IV* are presented.

The scheme enables data to be read into various arrangements of variables, with each routine starting to read on a new line of data and continuing until either a specified number of numbers has been read or an end of read symbol (/ or $) is encountered. In specifying the routines, the aim has been to give the correct interpretation of all well-formed numbers and, when ambiguous data is encountered, to give a reasonable interpretation and to issue a warning. A failure in the input routines can only occur from a system failure, for example if a number is too large for the computer to handle or an attempt is made to read beyond the end of a file.

A number of additional features have been introduced for convenience in using the routines. Alphanumeric comments can be inserted into the data by enclosing the characters between inverted commas (either single or double). An input variable can retain its previous values (that is, a 'null' datum is read) by the use of two consecutive commas, optionally enclosing blanks. If consecutive values being read are the same, then the data input can be simplified using the convention i*V to denote i consecutive occurrences of value V. The character / ends the read call forthwith: any further characters beyond / on the same line are ignored.

Whilst one of the aims in the specification of the routines has been to maintain compatibility with existing formatted data for Fortran programs, this has not always been possible. For example, in order to read the two numbers 149 and 736 in Fortran 2I3 format, they would appear in the data as 149736. The input scheme described here would interpret this as the single number 149736. For all cases where formatted data is separated by a non-digit character, the routines will read in the numbers as intended. In addition, the routines will interpret correctly free-format data prepared in accordance with the proposed Fortran 77 standard.

Details of the data input scheme are given in section 2, while the programs themselves are described in section 3 and listed in Appendix B. A simplified users' guide to the scheme appears as Appendix C.

Since this data input scheme was developed, the authors' attention has been drawn to a similar, but less comprehensive scheme, devised by D. Lovell which has been in use as part of the data handling software of the low-speed wind-tunnels at RAE Farnborough for a number of years.

2 DETAILS OF THE DATA INPUT SCHEME

The scheme provides routines for reading from one to nine real variables, one-dimensional, two-dimensional and three-dimensional real arrays, and similar routines for integer variables and arrays. The scheme is based on a routine FFORM which reads in a line of data as individual characters and then builds up the appropriate numerical values according to the rules proposed in section 2.1. In sections 2.2 to 2.6 various enhancements to this basic scheme are described.

2.1 Number recognition

The implementation of any system for recognising numbers requires a definition of a number. Such a definition should correspond with generally accepted interpretations and be as simple as possible. Here it is defined as any consecutive sequence of digits, which may be separated by a limited set of characters (digit separators) and may also be preceded by certain other characters (digit prefixes). The digit separators are a decimal point, or a letter E followed immediately by a digit, a space, a plus sign or a minus sign. The decimal point may come before the first digit (that is, be part of the prefix), and the characters after the E are interpreted as an exponent. Any number may have a plus or minus sign as its first character. These requirements are listed more exactly below by denoting any character by enclosing it in brackets, thus ⟨⟩. For example the character ⟨⟩ is the null character (that is, no character), whereas ⟨ ⟩ would indicate a space. Then with the following definitions,

digit = ⟨0⟩, ⟨1⟩, ⟨2⟩, ⟨3⟩, ⟨4⟩, ⟨5⟩, ⟨6⟩, ⟨7⟩, ⟨8⟩, ⟨9⟩

d = ⟨digit⟩, d⟨digit⟩ (*ie* d is a compact string of digits)

. = ⟨.⟩

$\bar{E}$ = ⟨E⟩, ⟨E ⟩, ⟨E+⟩, ⟨E−⟩

$\bar{S}$ = ⟨ ⟩, ⟨+⟩, ⟨−⟩

the numbers must be in one of the eight forms:

1. $\bar{S}d$
2. $\bar{S}.d$
3. $\bar{S}d.d$
4. $\bar{S}d.$
5. $\bar{S}d\bar{E}d$
6. $\bar{S}.d\bar{E}d$
7. $\bar{S}d.\bar{E}d$
8. $\bar{S}d.d\bar{E}d$

In the Fortran IV routines presented here, characters are read sequentially and a number is detected by the first digit found, and ended when the character list ceases to comply with one of the forms listed above.

2.2 Null data

A common requirement in inputting data is that an input variable should retain its previous value (that is, a null datum should be supplied). In the read routines described here, the occurrence of two commas optionally enclosing blanks is used to denote a null datum. Hence the character line: 123,, ,345 will give four values: 123, two null data and 345.

A comma occurring as the first non-blank character on a line implies a null datum, but a comma occurring as the last non-blank character on a line does not imply an extra null datum. Hence , 123,345, gives three values: 1 null datum, 123 and 345.

It should be noted that the new Fortran 77 free-format specification differs from the one used here, in that Fortran 77 interprets a comma occurring as the first non-blank character on a line as a null datum *only on the first line of data input* by a read statement. Hence two lines of data:

```
,123,345
,567
```

would be interpreted by the input routines described here as five values: 1 null datum, 123, 345, 1 null datum and 567, whereas Fortran 77 would interpret these characters as four values: 1 null datum, 123, 345 and 567.

2.3 Multiple data

Another common occurrence in inputting data is that several consecutive items in the data are identical. In this case, the data input can be simplified by using the convention i*V to denote i consecutive occurrences of the value V. Specifically, i must be a non-negative integer. Hence 3*3.14159 is

interpreted as three values of 3.14159 and 3*, is interpreted as three null data. When used in this way, the character * will not generate a warning (see section 2.6). Except in the case of multiple null data it is essential that the characters of i*V should be compact. For example, 3* 3.14159 will be interpreted as two values 3 and 3.14159. A similar interpretation will be made of 3.0*3.14159, or 3**3.14159 or 3 *3.14159.

If this convention is used to generate more data than is required, the input routine will end when the required amount of data has been read. For example, 100*1.5 can be used to fill a 60-element array with data. No warning is issued.

2.4 Comments

It will be appreciated that the definition of a number proposed in section 2.1 permits comments not containing digits to be included in the data. Hence for example 123,ABC,456 will be interpreted as 123 and 456.

As the facility to include alphanumeric comments in the data would seem to be of value, a convention is employed in which characters between inverted commas (either single or double) will be interpreted as a comment. Hence the characters

"AE1"
"AE2
'AE3"
'AE4'
'AE5

will be interpreted as comments but

AE6

will be interpreted as the value 6.

More specifically an alphanumeric comment is preceded by a ⟨"⟩ or ⟨'⟩ and ended by a second ⟨"⟩ or ⟨'⟩ or the end of a line.

2.5 End of read characters

While the user must specify the number of variables to be read by the routines (and normally the routines will continue reading until this number has been found) the character ⟨/⟩ will end the read call forthwith, any further characters beyond ⟨/⟩ on the same line, being ignored. Thus ⟨/⟩ can be used for the protection of further data or as a method of retaining the existing values for all further variables of the read call.

For all the routines except the two-dimensional and three-dimensional array routines, $ has the same effect as /. For two-dimensional arrays, $$ is needed to end the read while $ terminates the reading for the current row or column. Similarly with three-dimensional arrays, $$$ ends the read, while $ ends the current row or column and $$ ends the current row and column.

It should be noted that $ and / have no effect if contained within an alphanumeric comment (see section 2.4).

2.6 Warnings

Under certain conditions the routines generate warning messages. These messages are output to device 50, but can be suppressed altogether by setting the input parameter IDENT to zero. It should be noted that generation of a warning will not stop the program. However, even if the warning messages are suppressed, the output parameter IERR, which is normally positive, is set negative if a warning has been generated, so that the user may take appropriate action to stop the program automatically if he wishes.

Details of the warning messages are given in section 3.2, but the conditions for which they are generated are listed below. In the input scheme described here, the standard data delimiters are assumed to be ⟨ ⟩ or ⟨,⟩. Other characters can appear immediately adjacent to data, but will generate warning messages unless otherwise stated.

Specifically, if a number does not start at the beginning of a new line, is not immediately preceded by ⟨,⟩, ⟨ ⟩, ⟨"⟩, or ⟨'⟩ or is not immediately followed by ⟨,⟩, ⟨ ⟩, ⟨-⟩, ⟨+⟩, ⟨"⟩, ⟨'⟩, ⟨$⟩, ⟨/⟩ or the end of a line, a warning message locating the position of the offending character is generated. An exception is the character ⟨*⟩ when used to denote multiple data (see section 2.3). Similarly if an exponent contains other than ⟨ ⟩, ⟨+⟩, ⟨-⟩ or digits, a warning is generated.

In addition, if real data is encountered in a routine for reading integer values, the value is set to the nearest whole number and a warning is generated.

Examples of various acceptable data lines and ones which will generate warnings are given below.

(a) 123,456,789 or 123 456 789 will not generate a warning;

(b) 123"ABC"456 will not generate a warning, but 123ABC456 will generate two warnings;

(c) +3.78-2.63-1.97+2.96 will not generate a warning;

(d) 52*1.97 will not generate a warning since multiple data will be assumed, but 52**1.97 will generate two warnings;

(e) 1,3,4.2,5 , if read by an integer read routine will generate a warning and the third value will be set to 4.

3 THE PROGRAMS

The programs comprise 24 subroutines:

READ1,READ2,...,READ9	- read from one to nine real variables,
READI1,READI2,...,READI9	- read from one to nine integer variables,
READA	- read a one-dimensional real array,
READIA	- read a one-dimensional integer array,
READB	- read a two-dimensional real array,
READIB	- read a two-dimensional integer array,
READC	- read a three-dimensional real array,
READIC	- read a three-dimensional integer array.

In addition the following subroutines are used:

VAR9	- called by READ1,...,READ9. VAR9 reads nine real variables using READA,
IVAR9	- called by READI1,...,READI9. IVAR9 reads nine integer variables using READIA,
ARRAY	- called by READA, READB, READC, READIA, READIB, READIC. ARRAY reads real or integer variables into an array,
FFORM	- called by ARRAY. FFORM translates a line of characters into numerical values according to the rules proposed in section 2.

All the routines are written in standard Fortran IV. The characters (") and ('), while not included in the standard character set are available in Fortran on most computers and have been used in the routines. Within the routines it is assumed that characters are input from unit IUNIT and that warnings are printed on unit 50. Each "READ" statement should be associated with a whole number of lines of data (that is, data for two "READ" statements should not occur on the same line).

In the remainder of this section the routines are specified in more detail. The "READ" routines, together with VAR9 and IVAR9 are fairly straightforward in terms of logic and hence these are presented with a brief specification and listing. Subroutines ARRAY and FFORM are logically more complex and flow diagrams and annotated listings are included.

3.1 Specification of subroutines

3.1.1 READ1,READ2,...,READ9,READI1,READI2,...,READI9

Purpose

To read from one to nine real (READn) or integer (READIn) variables in free format.

Call statements

CALL READn(IUNIT,IDENT,IERR,X1,...Xn)
or
CALL READIn(IUNIT,IDENT,IERR,L1,...Ln)
where $1 \leqslant n \leqslant 9$.

Input parameters

IUNIT = unit identification for data input

IDENT = statement identifier and warning control:
If IDENT = 0, no warnings are printed;
If IDENT $\geqslant$ 1, identified warnings are printed;
If IDENT $\geqslant$ 1000, all data are printed too.

Output parameters

IERR = number of values read x sgn
where sgn = 1 if there are no warnings,
= -1 if there are warnings.

X1,...Xn or L1,...Ln ($1 \leqslant n \leqslant 9$) are set to the values read in.

Subroutines called

READn uses VAR9, READA, ARRAY and FFORM,
READIn uses IVAR9, READIA, ARRAY and FFORM.

Exit

The routines end when either n values have been read in or a ⟨/⟩ or a ⟨$⟩ is encountered.

3.1.2 READA, READIA

Purpose

To read a one-dimensional real (READA) or integer (READIA) array.

Call statements

CALL READA(IUNIT,IDENT,IERR,A,I1,I2)
or
CALL READIA(IUNIT,IDENT,IERR,IA,I1,I2)

Input parameters

IUNIT and IDENT are defined in section 3.1.1;

A is a one-dimensional real array of dimension $\geqslant$I2,
IA is a one-dimensional integer array of dimension $\geqslant$I2,

I1, I2 are the initial and final subscripts for which data is read.

Output parameters

IERR is defined in section 3.1.1.
The values read in are stored in A or IA.

Subroutines called

Both routines call ARRAY and FFORM.

Exit

The routines end when either I2-I1+1 values have been read in or a ⟨/⟩ or a ⟨$⟩ is encountered.

3.1.3 READB, READIB

Purpose — To read a two-dimensional real (READB) or integer (READIB) array.

Call statements —
CALL READB(IUNIT,IDENT,IERR,B,I1,I2,NI,J1,J2,NJ,IØRD)
or
CALL READIB(IUNIT,IDENT,IERR,IB,I1,I2,NI,J1,J2,NJ,IØRD)

Input parameters — IUNIT and IDENT are defined in section 3.1.1,
I1, I2 are the initial and final values of I, and
J1, J2 are the initial and final values of J for which data is required in the two-dimensional arrays B(I,J) or IB(I,J).
NI, NJ are the dimensions I and J respectively of B or IB.
IORD denotes the order in which the array is filled. If IORD = 21, the subscript I varies while J remains fixed (columns are read in matrix terms). Otherwise J varies while I remains fixed (rows are read in matrix terms).

Output parameters — IERR is defined in section 3.1.1.
The values read in are stored in B or IB.

Subroutines called — Both routines call ARRAY and FFORM.

Exit — The routines end when (I2-I+1)*(J2-J1+1) values have been read in, or when a ⟨/⟩ or two consecutive ⟨$⟩ characters are encountered. In these routines, a single ⟨$⟩ denotes the end of a sequence of the more rapidly varying subscript. It should be noted, however, that the appearance of a ⟨/⟩ or ⟨$⟩ on a line causes further data on that line to be ignored. Hence two sequences terminated by ⟨$⟩ should not appear on the same line.

3.1.4 READC, READIC

Purpose — To read a three-dimensional real (READC) or integer (READIC) array.

Call statements —
CALL READC(IUNIT,IDENT,IERR,C,I1,I2,NI,J1,J2,NJ,K1,K2,NK,IØRD)
or
CALL READIC(IUNIT,IDENT,IERR,IC,I1,I2,NI,J1,J2,NJ,K1,K2,NK,IØRD)

Input parameters — IUNIT and IDENT are defined in section 3.1.1,
I1, I2 are the initial and final values of I,
J1, J2 are the initial and final values of J, and
K1, K2 are the initial and final values of K for which data is required in the three-dimensional arrays C(I,J,K) or IC(I,J,K). NI, NJ, NK are the dimensions I, J and K respectively of arrays C or IC.
IØRD denotes the order in which the array is filled. IØRD = 132 means that J (identified with 2) varies most rapidly, followed by K (identified with 3), followed by I (identified with 1). Other significant values of IØRD are as follows:

IΘRD = 231 implies JKI order,
= 213 implies JIK order,
= 312 implies KIJ order,
= 321 implies KJI order,

For all other values of IΘRD, the order is assumed to be IJK.

Output parameters — IERR is defined in section 3.1.1.
The values read in are stored in C or IC.

Subroutines called — Both routines call ARRAY and FFORM.

Exit — The routines end when (I2-I1+1)*(J2-J1+1)*(K2-K1+1) values have been read in or a ⟨/⟩ or three consecutive ⟨$⟩'s are encountered. In this routine a single ⟨$⟩ denotes the end of a sequence of the most rapidly varying subscript and two consecutive ⟨$⟩ characters denote the end of a two-dimensional sub-array with the two most rapidly varying subscripts.

3.1.5 VAR9, IVAR9 (subsidiary routines)

Purpose — To read N real (VAR9) or integer (IVAR9) variables.

Call statements — CALL VAR9(IUNIT,IDENT,IERR,N,X1,...,X9)
or
CALL IVAR9(IUNIT,IDENT,IERR,N,L1,...,L9)

Input parameters — IUNIT and IDENT are defined in section 3.1.1.
N is the number of variables required $1 \leqslant N \leqslant 9$.

Output parameters — IERR is defined in section 3.1.1.
X1,...XN or L1,...,LN are the values found

Subroutines called — VAR9 uses READA, ARRAY and FFORM,
IVAR9 uses READIA, ARRAY and FFORM.

Exit — The routines end when either N values have been read in or a ⟨/⟩ or a ⟨$⟩, is encountered.

3.1.6 ARRAY (subsidiary routine)

Purpose — To read data into a three-dimensional real or integer array.

Call statement — CALL ARRAY(IUNIT,IDENT,IERR,C,NI,NJ,NK,IC,MI,MJ,MK,LIM,
JΘRD,IDIM,ININT)

Input parameters — IUNIT, IDENT are defined in section 3.1.1

ININT = 1 if ARRAY is called by an integer "READ" routine
= 0 otherwise

NI, NJ, NK are the dimensions of a real array C(I,J,K)
MI, MJ, MK are the dimensions of an integer array IC(I,J,K)

IDIM is the dimension of array in "READ" statement
(=1 for reading variables)

LIM(6) is an array containing initial and final values of I, J, K for which data is required

JØRD indicates the order in which the array is filled (see section 3.1.4).

Output parameters — IERR is defined in section 3.1.1.
The values read in are stored in IC if ININT = 1 or C otherwise.

Subroutines called — FFORM

Exit — The routine ends when the required number of data has been read, when ⟨/⟩ is found or when ⟨$⟩ is encountered for IDIM = 1, ⟨$$⟩ is found for IDIM = 2, ⟨$$$⟩ is found for IDIM = 3.

An annotated listing of ARRAY appears in Appendix B and a flow diagram is given in Appendix A.

3.1.7 FFORM (subsidiary routine)

Purpose — To read a line of characters and translate into numerical values.

Call statement — CALL FFORM(IUNIT,IDENT,NREC,INEND,INWAR,NUM,AA,JDAT,ININT)

Input parameters — IUNIT and IDENT are defined in section 3.1.1
NREC is the data record counter
INWAR is the number of warnings generated in current read routine
ININT = 1 for reading integers
= 0 otherwise

Output parameters — AA(40) is an array for storage of numbers found

INEND = 0 if no end of read character has been found
1 if ⟨$⟩ has occurred
2 if ⟨$$⟩ has occurred
3 if ⟨$$$⟩ has occurred
4 if ⟨/⟩ has occurred.

INWAR is the number of warnings generated in current read routine
JDAT(40) is an array to indicate if the corresponding value in AA is normal (JDAT = 1), null (= 0) or multiple (= 2)
NUM is the number of values found.

Subroutines called — None

Exit — The routine ends when the end of a line is reached.

An annotated listing of FFORM is given in Appendix B and a flow diagram appears in Appendix A.

3.2 Warning messages

The conditions for which warning messages are generated are defined in section 2.6. The warning messages are normally printed on device 50, but can be suppressed by setting the parameter IDENT to zero.

The first warning condition found in a routine gives the following heading:

**** DATA READ WARNINGS ****

Then one or more of the following warning messages will be given:

(a) FOR IDENT = I, AFTER J LINES & K CHAR C_1C_2 message,
where message is either STARTS NUMBER ,
or ENDS NUMBER ,
or IN EXPONENT .

Here I = identifier
J = number of lines of data read in
K = character count on current line
C_1C_2 are the two characters in positions K-1 and K .

(b) FOR IDENT = I , V FOUND, INTEGER N ASSUMED,
where I = identifier
V = floating point value
N = nearest integer value to V .

If IDENT $\geqslant$ 1000 and a warning message has been given, the line of characters is printed out followed by a blank line with the character + showing the positions of the characters generating any warnings.

3.3 Implementation

The free-format data input scheme presented here has been implemented and tested on a number of computer systems. Up until now the routines have been used successfully on the following computers:

1) DEC KL10B at RAE Farnborough,
2) IBM 370/168 at UKAEA Harwell,
3) ICL 1906S at RAE Farnborough,
4) ICL 4130 at RAE Bedford,
5) Honeywell Mk III Time Sharing Service.

The only modification found to be necessary was on the ICL 4130, where the double quotation character (") is not allowed in Fortran programs. For use on the 4130,

therefore, all references to the variable IDQ in routine FFORM should be deleted. (The inclusion of alphanumeric comments can still be achieved, of course, by the use of single quotes ⟨'⟩.)

The core requirements for the data input scheme depend on how many of the 'READ' routines are called. For example, each of the individual READn routines requires approximately 40 words, but the call will also involve READA (80 words), VAR9 (200 words), ARRAY (600 words) and FFORM (1800 words). Calling all 24 'READ' routines would need approximately 4K, but, for typical usage, involving say six of the routines the storage requirement would amount to 3K.

Experience has shown that the 'READ' routines are between 10 and 15 times slower than using the normal Fortran formatted input. Typically, on the DEC KL10, 1000 numbers can be input in less than 10 seconds.

On the RAE DEC KL10B, the read system is implemented for the users' convenience in SYS: RAE LIB. A help file HELP FOREAD gives details, and a program called by RUN READEM gives a demonstration of the system. In practical use on the DEC, the read system has been found to be reliable and very accommodating. The error warnings on the DEC are arranged to appear on device 5, that is at the user's teletype or in the LOG file for batch runs.

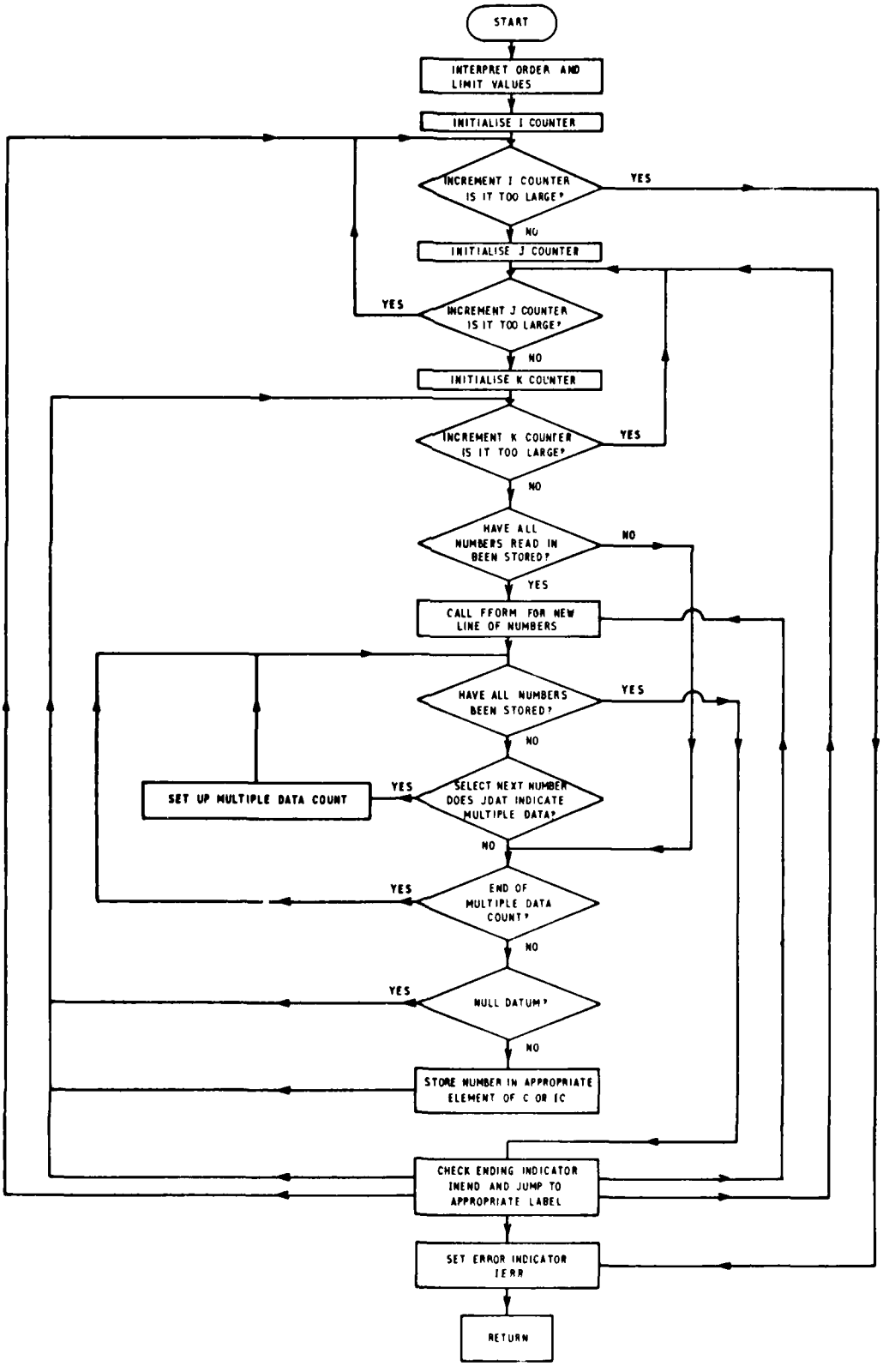
4 CONCLUDING REMARKS

Routines, written in Fortran IV, for the input of numerical data in free-format have been described. Provided a set of simple rules are obeyed, the scheme gives the user the flexibility to format data in the most convenient way. Additional features include provision for input variables to retain their previous values (null data), for identical consecutive data to be described in a simplified form (multiple data) and for alphanumeric comments to appear amongst the data.

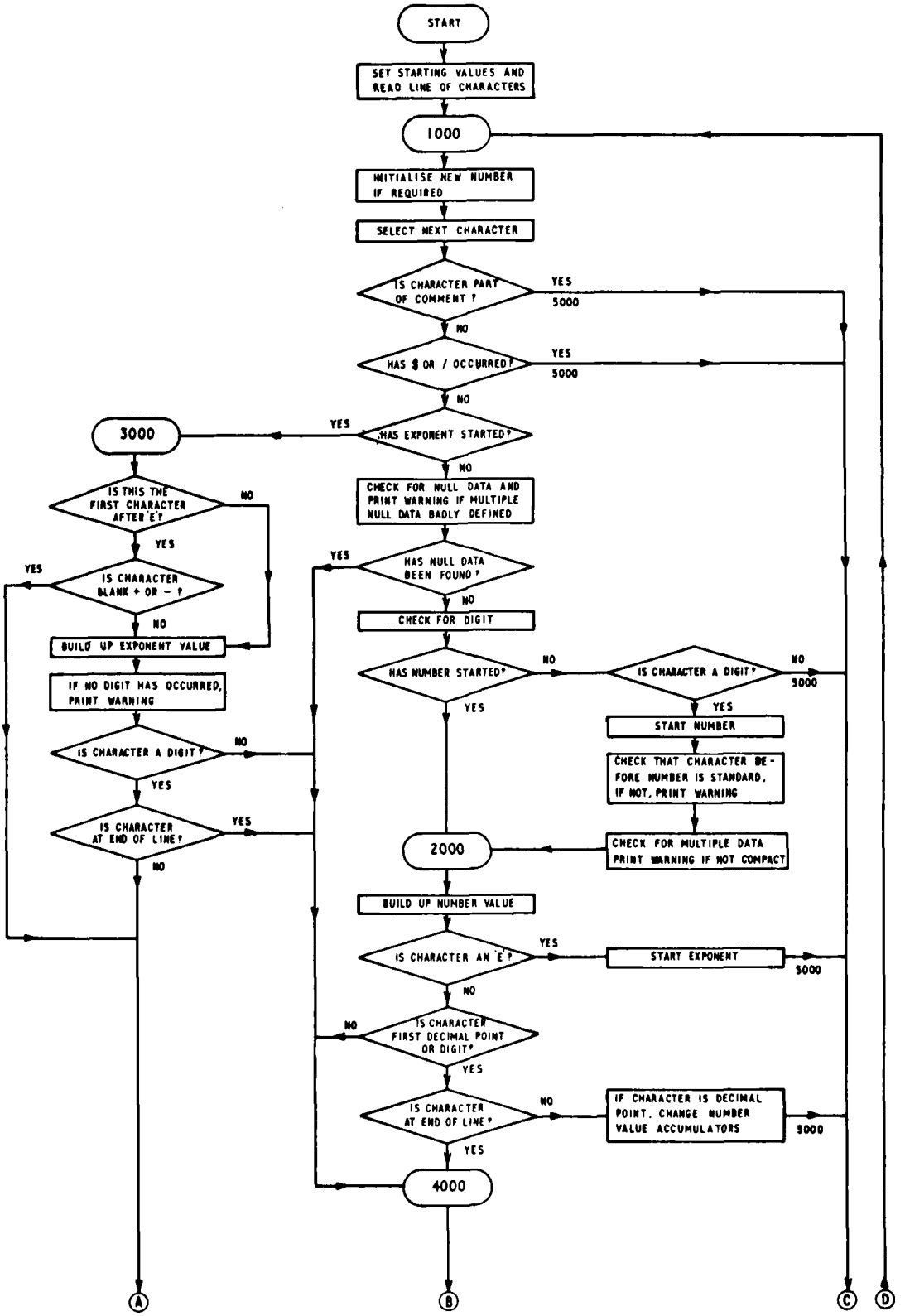
The input scheme was devised primarily to simplify the reading of data into large flow-field calculation programs which are being developed at RAE for use throughout the UK aircraft industry. For this type of program, the extra core requirements and time taken when reading in data in this way (typically less than 4K and less than 10 seconds per 1000 numbers on the DEC KL10B) are very small compared with the resources used by the program as a whole (typically 64K and 30 minutes running time). The use of standard Fortran IV throughout the routines means that programs using this input scheme can be transferred easily from one computer to another.

Appendix A

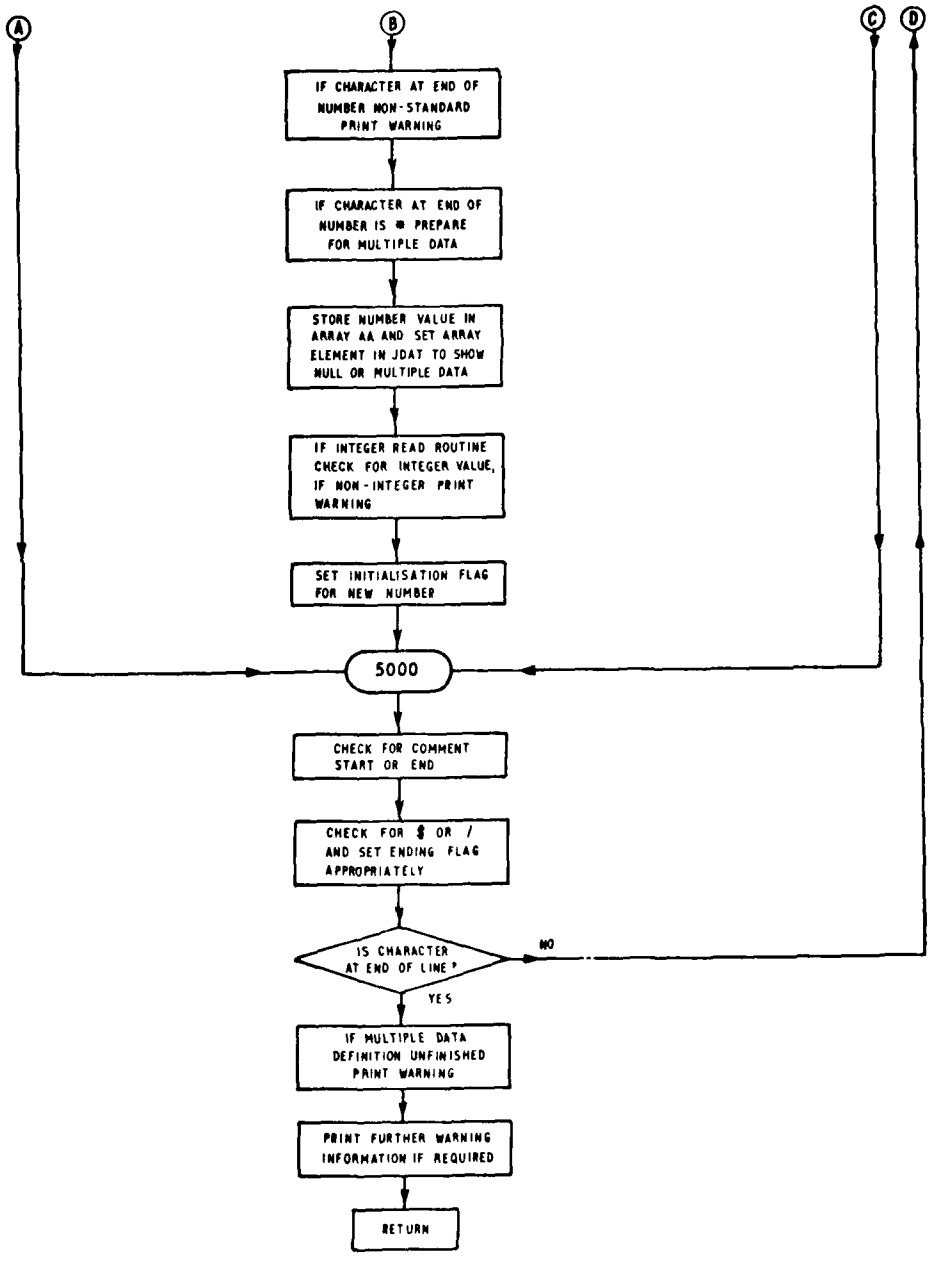
FLOW DIAGRAMS



FLOW DIAGRAM OF SUBROUTINE FFORM (1)



FLOW DIAGRAM OF SUBROUTINE FFORM (2)



Appendix B

LISTINGS

```
0001  C*************************************************************************
0002  C
0003  C          FORTRAN FREE-FORMAT DATA INPUT SCHEME
0004  C
0005  C          1 : READ1 TO READ9 & READ11 TO READ19
0006  C
0007  C          G BUTLER & J PIKE : MAY '73
0008  C
0009  C*************************************************************************
0010  C
0011  C          THESE ROUTINES ALLOW FROM 1 TO 9 REAL OR INTEGER
0012  C          VARIABLES TO BE READ
0013  C
0014  C
0015  C          INPUT PARAMETERS
0016  C          ----------------
0017  C
0018  C          IDENT = IDENTIFIER AND WARNING CONTROL
0019  C                  IF IDENT = 0, NO WARNINGS ARE PRINTED
0020  C                  IF IDENT.GE.1, IDENTIFIED WARNINGS ARE PRINTED ON
0021  C                  DEVICE 50
0022  C                  IF IDENT.GE.1000, ALL DATA ARE PRINTED ON DEVICE 50
0023  C          IUNIT = DATA INPUT UNIT
0024  C
0025  C
0026  C          OUTPUT PARAMETERS
0027  C          -----------------
0028  C
0029  C          IERR = ERROR INDICATOR
0030  C               = NUMBER OF VALUES READ MULTIPLIED BY SGN
0031  C                 WHERE SGN = 1 IF NO WARNINGS HAVE OCCURRED
0032  C                       SGN = -1 IF WARNINGS HAVE OCCURRED
0033  C          X1,.....,XM OR L1,.....,LM (1.LE.M.LE.9) ARE SET TO THE
0034  C             VALUES FOUND
0035  C
0036  C
0037  C*************************************************************************
0038  C
0039  C
0040        SUBROUTINE READ1(IUNIT,IDENT,IERR,X1)
0041        CALL VAR9(IUNIT,IDENT,IERR,1,X1,X2,X3,X4,X5,X6,X7,X8,X9)
0042        RETURN
0043        END
0044        SUBROUTINE READ2(IUNIT,IDENT,IERR,X1,X2)
0045        CALL VAR9(IUNIT,IDENT,IERR,2,X1,X2,X3,X4,X5,X6,X7,X8,X9)
0046        RETURN
0047        END
0048        SUBROUTINE READ3(IUNIT,IDENT,IERR,X1,X2,X3)
0049        CALL VAR9(IUNIT,IDENT,IERR,3,X1,X2,X3,X4,X5,X6,X7,X8,X9)
0050        RETURN
0051        END
0052        SUBROUTINE READ4(IUNIT,IDENT,IERR,X1,X2,X3,X4)
0053        CALL VAR9(IUNIT,IDENT,IERR,4,X1,X2,X3,X4,X5,X6,X7,X8,X9)
0054        RETURN
```

```
0055          END
0056          SUBROUTINE READ5(IUNIT,IDENT,IERR,X1,X2,X3,X4,X5)
0057          CALL VAR9(IUNIT,IDENT,IERR,5,X1,X2,X3,X4,X5,X6,X7,X8,X9)
0058          RETURN
0059          END
0060          SUBROUTINE READ6(IUNIT,IDENT,IERR,X1,X2,X3,X4,X5,X5)
0061          CALL VAR9(IUNIT,IDENT,IERR,6,X1,X2,X3,X4,X5,X6,X7,X8,X9)
0062          RETURN
0063          END
0064          SUBROUTINE READ7(IUNIT,IDENT,IERR,X1,X2,X3,X4,X5,X5,X7)
0065          CALL VAR9(IUNIT,IDENT,IERR,7,X1,X2,X3,X4,X5,X6,X7,X8,X9)
0066          RETURN
0067          END
0068          SUBROUTINE READ8(IUNIT,IDENT,IERR,X1,X2,X3,X4,X5,X5,X7,X8)
0069          CALL VAR9(IUNIT,IDENT,IERR,8,X1,X2,X3,X4,X5,X6,X7,X8,X9)
0070          RETURN
0071          END
0072          SUBROUTINE READ9(IUNIT,IDENT,IERR,X1,X2,X3,X4,X5,X5,X7,X8,X9)
0073          CALL VAR9(IUNIT,IDENT,IERR,9,X1,X2,X3,X4,X5,X6,X7,X8,X9)
0074          RETURN
0075          END
0076    C
0077          SUBROUTINE READI1(IUNIT,IDENT,IERR,L1)
0078          CALL IVAR9(IUNIT,IDENT,IERR,1,L1,L2,L3,L4,L5,L6,L7,L8,L9)
0079          RETURN
0080          END
0081          SUBROUTINE READI2(IUNIT,IDENT,IERR,L1,L2)
0082          CALL IVAR9(IUNIT,IDENT,IERR,2,L1,L2,L3,L4,L5,L6,L7,L8,L9)
0083          RETURN
0084          END
0085          SUBROUTINE READI3(IUNIT,IDENT,IERR,L1,L2,L3)
0086          CALL IVAR9(IUNIT,IDENT,IERR,3,L1,L2,L3,L4,L5,L6,L7,L8,L9)
0087          RETURN
0088          END
0089          SUBROUTINE READI4(IUNIT,IDENT,IERR,L1,L2,L3,L4)
0090          CALL IVAR9(IUNIT,IDENT,IERR,4,L1,L2,L3,L4,L5,L6,L7,L8,L9)
0091          RETURN
0092          END
0093          SUBROUTINE READI5(IUNIT,IDENT,IERR,L1,L2,L3,L4,L5)
0094          CALL IVAR9(IUNIT,IDENT,IERR,5,L1,L2,L3,L4,L5,L6,L7,L8,L9)
0095          RETURN
0096          END
0097          SUBROUTINE READI6(IUNIT,IDENT,IERR,L1,L2,L3,L4,L5,L6)
0098          CALL IVAR9(IUNIT,IDENT,IERR,6,L1,L2,L3,L4,L5,L6,L7,L8,L9)
0099          RETURN
0100          END
0101          SUBROUTINE READI7(IUNIT,IDENT,IERR,L1,L2,L3,L4,L5,L6,L7)
0102          CALL IVAR9(IUNIT,IDENT,IERR,7,L1,L2,L3,L4,L5,L6,L7,L8,L9)
0103          RETURN
0104          END
0105          SUBROUTINE READI8(IUNIT,IDENT,IERR,L1,L2,L3,L4,L5,L6,L7,L8)
0106          CALL IVAR9(IUNIT,IDENT,IERR,8,L1,L2,L3,L4,L5,L6,L7,L8,L9)
0107          RETURN
0108          END


0109          SUBROUTINE READI9(IUNIT,IDENT,IERR,L1,L2,L3,L4,L5,L6,L7,L8,L9)
0110          CALL IVAR9(IUNIT,IDENT,IERR,9,L1,L2,L3,L4,L5,L6,L7,L8,L9)
0111          RETURN
0112          END
**** LIST END ****
```

```
0031  C*****************************************************************
0032  C
0033  C         FORTRAN FREE-FORMAT DATA INPUT SCHEME
0034  C
0035  C         2 : READA, READIA, READB, READIB, READC & READIC
0036  C
0037  C         G BUTLER & J PIKE : MAY '73
0038  C
0039  C*****************************************************************
0040  C
0041  C         READA READS A ONE-DIMENSIONAL REAL ARRAY
0042  C         READB READS A TWO-DIMENSIONAL REAL ARRAY
0043  C         READC READS A THREE-DIMENSIONAL REAL ARRAY
0044  C         READIA READS A ONE-DIMENSIONAL INTEGER ARRAY
0045  C         READIB READS A TWO-DIMENSIONAL INTEGER ARRAY
0046  C         READIC READS A THREE-DIMENSIONAL INTEGER ARRAY
0047  C
0048  C
0049  C         INPUT PARAMETERS
0050  C         ----------------
0051  C
0052  C         IDENT = IDENTIFIER AND WARNING CONTROL
0053  C         IUNIT = DATA INPUT UNIT
0054  C         A IS A ONE-DIMENSIONAL REAL ARRAY OF DIMENSION .GE.I2
0055  C         B IS A TWO-DIMENSIONAL REAL ARRAY OF DIMENSION NI,NJ
0056  C         C IS A THREE-DIMENSIONAL REAL ARRAY OF DIMENSION NI,NJ,NK
0057  C         IA IS A ONE-DIMENSIONAL INTEGER ARRAY OF DIMENSION .GE.I2
0058  C         IB IS A TWO-DIMENSIONAL INTEGER ARRAY OF DIMENSION NI,NJ
0059  C         IC IS A THREE-DIMENSIONAL INTEGER ARRAY OF DIMENSION NI,NJ,NK
0060  C         I1,I2 ARE THE INITIAL AND FINAL VALUES OF I,
0061  C         J1,J2 ARE THE INITIAL AND FINAL VALUES OF J,
0062  C         K1,K2 ARE THE INITIAL AND FINAL VALUES OF K,
0063  C               FOR WHICH DATA IS TO BE READ
0064  C         IORD DENOTES THE ORDER IN WHICH THE APPROPRIATE ARRAY IS FILLED
0065  C
0066  C
0067  C         OUTPUT PARAMETERS
0068  C         -----------------
0069  C
0070  C         IERR = ERROR INDICATOR
0071  C         THE VALUES ARE STORED IN THE APPROPRIATE ARRAY
0072  C
0073  C
0074  C*****************************************************************
0075  C
0076  C
0077        SUBROUTINE READA(IUNIT,IDENT,IERR,A,I1,I2)
0078        DIMENSION A(I2),IA(1),LIM(6)
0079        DO 1 K=1,6
0080      1 LIM(K)=1
0081        LIM(3)=I1
0082        LIM(6)=I2
0083        JORD=123
0084        CALL ARRAY(IUNIT,IDENT,IERR,A,1,1,I2,IA,1,1,1,LIM,JORD,1,0)
```

```
      RETURN
      END
C
      SUBROUTINE READIA(IUNIT,IDENT,IERR,IA,I1,I2)
      DIMENSION IA(I2),A(1),LIM(6)
      DO 1 K=1,6
    1 LIM(K)=1
      LIM(3)=I1
      LIM(6)=I2
      JORD=123
      CALL ARRAY(IUNIT,IDENT,IERR,A,1,1,1,IA,1,1,I2,LIM,JORD,1,1)
      RETURN
      END
C
      SUBROUTINE READB(IUNIT,IDENT,IERR,B,I1,I2,NI,J1,J2,NJ,IORD)
      DIMENSION B(NI,NJ),IB(1),LIM(6)
      LIM(1)=1
      LIM(2)=I1
      LIM(3)=J1
      LIM(4)=1
      LIM(5)=I2
      LIM(6)=J2
      JORD=123
      IF(IORD.EQ.21) JORD=132
      CALL ARRAY(IUNIT,IDENT,IERR,B,1,NI,NJ,IB,1,1,1,LIM,JORD,2,0)
      RETURN
      END
C
      SUBROUTINE READIB(IUNIT,IDENT,IERR,IB,I1,I2,NI,J1,J2,NJ,IORD)
      DIMENSION IB(NI,NJ),B(1),LIM(6)
      LIM(1)=1
      LIM(2)=I1
      LIM(3)=J1
      LIM(4)=1
      LIM(5)=I2
      LIM(6)=J2
      JORD=123
      IF(IORD.EQ.21) JORD=132
      CALL ARRAY(IUNIT,IDENT,IERR,B,1,1,1,IB,1,NI,NJ,LIM,JORD,2,1)
      RETURN
      END
C
      SUBROUTINE READC(IUNIT,IDENT,IERR,C,I1,I2,NI,J1,J2,NJ,
     1    K1,K2,NK,IORD)
      DIMENSION C(NI,NJ,NK),IC(1),LIM(6)
      LIM(1)=I1
      LIM(2)=J1
      LIM(3)=K1
      LIM(4)=I2
      LIM(5)=J2
      LIM(6)=K2
      JORD=123
      IF(IORD.EQ.132) JORD=132
      IF(IORD.EQ.231) JORD=231
```

```
0109        IF(IORD.EQ.213) JORD=213
0110        IF(IORD.EQ.312) JORD=312
0111        IF(IORD.EQ.321) JORD=321
0112        CALL ARRAY(IUNIT,IDENT,IERR,C,NI,NJ,NK,IC,1,1,1,LIM,JORD,3,0)
0113        RETURN
0114        END
0115  C
0116        SUBROUTINE READIC(IUNIT,IDENT,IERR,IC,I1,I2,NI,J1,J2,NJ,
0117       1    K1,K2,NK,IORD)
0118        DIMENSION IC(NI,NJ,NK),LIM(6),C(1)
0119        LIM(1)=I1
0120        LIM(2)=J1
0121        LIM(3)=K1
0122        LIM(4)=I2
0123        LIM(5)=J2
0124        LIM(6)=K2
0125        JORD=123
0126        IF(IORD.EQ.132) JORD=132
0127        IF(IORD.EQ.231) JORD=231
0128        IF(IORD.EQ.213) JORD=213
0129        IF(IORD.EQ.312) JORD=312
0130        IF(IORD.EQ.321) JORD=321
0131        CALL ARRAY(IUNIT,IDENT,IERR,C,1,1,1,IC,NI,NJ,NK,LIM,JORD,3,1)
0132        RETURN
0133        END
**** LIST END ****
```

```
0001  C*****************************************************************************
0002  C
0003  C          FORTRAN FREE-FORMAT DATA INPUT SCHEME
0004  C
0005  C          3 : VAR9 & IVAR9
0006  C
0007  C          G BUTLER & J PIKE : MAY '73
0008  C
0009  C*****************************************************************************
0010  C
0011  C          VAR9 READS N REAL VARIABLES (1.LE.N.LE.9)
0012  C          IVAR9 READS N INTEGER VARIABLES (1.LE.N.LE.9)
0013  C
0014  C
0015  C          INPUT PARAMETERS
0016  C          ----------------
0017  C
0018  C          IDENT = IDENTIFIER & WARNING CONTROL
0019  C          IUNIT = DATA INPUT UNIT
0020  C          N = NUMBER OF VARIABLES REQUIRED
0021  C
0022  C
0023  C          OUTPUT PARAMETERS
0024  C          -----------------
0025  C
0026  C          IERR = ERROR-INDICATOR
0027  C          X1,X2,.....,XN OR L1,L2,.....,LN ARE THE VALUES FOUND
0028  C             WHERE (1.LE.N.LE.9)
0029  C
0030  C
0031  C*****************************************************************************
0032  C
0033  C
0034        SUBROUTINE VAR9(IUNIT,IDENT,IERR,N,X1,X2,X3,X4,X5,X6
0035       1     ,X7,X8,X9)
0036        DIMENSION AA(9)
0037        AA(1)=X1
0038        AA(2)=X2
0039        AA(3)=X3
0040        AA(4)=X4
0041        AA(5)=X5
0042        AA(6)=X6
0043        AA(7)=X7
0044        AA(8)=X8
0045        AA(9)=X9
0046        CALL READ9(IUNIT,IDENT,IERR,AA,1,N)
0047        X1=AA(1)
0048        X2=AA(2)
0049        X3=AA(3)
0050        X4=AA(4)
0051        X5=AA(5)
0052        X6=AA(6)
0053        X7=AA(7)
0054        X8=AA(8)
```

```
0055        X9=AA(9)
0056        RETURN
0057        END
0058  C
0059        SUBROUTINE IVAR9(IUNIT,IDENT,IERR,N,L1,L2,L3,L4,L5,L6
0060       1    ,L7,L8,L9)
0061        DIMENSION LA(9)
0062        LA(1)=L1
0063        LA(2)=L2
0064        LA(3)=L3
0065        LA(4)=L4
0066        LA(5)=L5
0067        LA(6)=L6
0068        LA(7)=L7
0069        LA(8)=L8
0070        LA(9)=L9
0071        CALL REDIA(IUNIT,IDENT,IERR,LA,1,N)
0072        L1=LA(1)
0073        L2=LA(2)
0074        L3=LA(3)
0075        L4=LA(4)
0076        L5=LA(5)
0077        L6=LA(6)
0078        L7=LA(7)
0079        L8=LA(8)
0080        L9=LA(9)
0081        RETURN
0082        END
**** LIST END ****
```

```
0001  C*************************************************************************
0002  C
0003  C         FORTRAN FREE-FORMAT DATA INPUT SCHEME
0004  C
0005  C         4 : ARRAY
0006  C
0007  C         G BUTLER & J PIKE : MAY '73
0008  C
0009  C*************************************************************************
0010  C
0011  C         ARRAY READS DATA INTO THE THREE-DIMENSIONAL REAL ARRAY C OR
0012  C         THE THREE-DIMENSIONAL INTEGER ARRAY IC , ACCORDING TO THE
0013  C         VALUE OF THE PARAMETER ININT
0014  C
0015  C
0016  C         INPUT PARAMETERS
0017  C         ----------------
0018  C
0019  C         IDENT = IDENTIFIER & WARNING CONTROL
0020  C         IUNIT = DATA INPUT UNIT
0021  C         NI,NJ,NK = DIMENSIONS OF REAL ARRAY C
0022  C         MI,MJ,MK = DIMENSIONS OF INTEGER ARRAY IC
0023  C         IDIM = DIMENSION OF ARRAY IN CALLING ROUTINE
0024  C         ININT = 1 FOR READING INTEGERS , 0 OTHERWISE
0025  C         LIM(6) = ARRAY CONTAINING INITIAL AND FINAL
0026  C                  VALUES OF I,J & K FOR WHICH DATA IS REQUIRED
0027  C         JORD = ORDER IN WHICH THE ARRAY IS FILLED
0028  C
0029  C
0030  C         OUTPUT PARAMETERS
0031  C         -----------------
0032  C
0033  C         IERR= ERROR INDICATOR
0034  C         VALUES READ IN ARE STORED IN IC IF ININT= 1,
0035  C         OR C OTHERWISE.
0036  C
0037  C
0038  C         OTHER PARAMETERS
0039  C         ----------------
0040  C
0041  C         AA(40) = ARRAY FOR NUMBER RETURN FROM FFORM
0042  C         INEND = ENDING INDICATOR
0043  C         INWAR = WARNING INDICATOR
0044  C         JDAT(40) = ARRAY TO INDICATE IF CORRESPONDING AA VALUE
0045  C                    IS NULL OR MULTIPLE
0046  C         KM = MULTIPLE DATA COUNTER
0047  C         KN = COUNTER FOR STORING CURRENT LINE
0048  C         MULT = MULTIPLIER FOR MULTIPLE DATA
0049  C         NCOUNT = COUNTER FOR NUMBERS FOUND
0050  C         NREC = DATA LINE COUNTER
0051  C         NUM = NUMBER OF VALUES FOUND ON CURRENT LINE
0052  C
0053  C
0054  C
```

```
0055  C          ICOUNT, JCOUNT, KCOUNT REFER TO A " NATURAL "
0056  C             COUNTING ORDER, WITH K VARYING MORE RAPIDLY
0057  C             THAN J, WHICH IN TURN VARIES MORE RAPIDLY THAN
0058  C             I. THE ICOUNT, JCOUNT, KCOUNT VALUES FOR ANY
0059  C             DATA ITEM ARE THEN RELATED TO ITS ADDRESS IN
0060  C             ARRAY C OR IC USING II, JJ, KK DERIVED FROM THE
0061  C             ORDER PARAMETER JORD.
0062  C
0063  C
0064  C*************************************************************
0065  C
0066  C
0067        SUBROUTINE ARRAY(IUNIT,IDENT,IERR,C,NI,NJ,NK,IC,MI,MJ,MK,
0068       1    LIM,JORD,IDIM,INCNT)
0069        DIMENSION C(NI,NJ,NK),IC(MI,MJ,MK),IHED(40),IDAT(40),LIM(6)
0070  C.....................................................................
0071  C                                        INTERPRET ORDER AND LIMIT VALUES.
0072        II=JORD/100
0073        JJ=(JORD-100*II)/10
0074        KK=JORD-100*II-10*JJ
0075        NI1=LIM(II)
0076        NI2=LIM(II+3)
0077        NJ1=LIM(JJ)
0078        NJ2=LIM(JJ+3)
0079        NK1=LIM(KK)
0080        NK2=LIM(KK+3)
0081        NUM=0
0082        INWAR=0
0083        KN=1
0084        MULT=0
0085        KM=0
0086        NREC=0
0087        NCOUNT=0
0088  C.....................................................................
0089  C                                        INITIALISE I COUNTER.
0090        ICOUNT=NI1-1
0091  C.....................................................................
0092  C                                        INCREMENT I COUNTER AND CHECK
0093  C                                        IF TOO LARGE?
0094     11 CONTINUE
0095        ICOUNT=ICOUNT+1
0096        IF(ICOUNT.GT.NI2) GO TO 21
0097  C.....................................................................
0098  C                                        INITIALISE J COUNTER.
0099        JCOUNT=NJ1-1
0100  C.....................................................................
0101  C                                        INCREMENT J COUNTER AND CHECK
0102  C                                        IF TOO LARGE ?
0103     12 CONTINUE
0104        JCOUNT=JCOUNT+1
0105        IF(JCOUNT.GT.NJ2) GO TO 11
0106  C.....................................................................
0107  C                                        INITIALISE K COUNTER.
0108        KCOUNT=NK1-1
```

```
0109  C.....................................................................
0110  C                                        INCREMENT K COUNTER AND CHECK
0111  C                                        IF TOO LARGE ?
0112     13 CONTINUE
0113        KCOUNT=KCOUNT+1
0114        IF(KCOUNT.GT.NK2) GO TO 12
0115  C.....................................................................
0116  C                                        HAVE ALL NUMBERS READ IN
0117  C                                        BEEN STORED ?
0118        IF(KN.LE.NUM) GO TO 17
0119  C.....................................................................
0120  C                                        CALL FFORM FOR NEW NUMBERS.
0121     14 CONTINUE
0122        NREC=NREC+1
0123        CALL FFORM(IUNIT,IDENT,NREC,INEND,INWAR,NUM,AA,JDAT,ININT)
0124        KN=0
0125  C.....................................................................
0126  C                                        HAVE ALL NUMBERS READ IN
0127  C                                        BEEN STORED ?
0128     15 CONTINUE
0129        MULT=1
0130        KM=0
0131     16 CONTINUE
0132        KN=KN+1
0133        IF(KN.GT.NUM) GO TO 19
0134  C.....................................................................
0135  C                                        SELECT NEXT NUMBER AND CHECK
0136  C                                        FOR MULTIPLE DATA.
0137        IF(JDAT(KN).NE.2) GO TO 17
0138  C.....................................................................
0139  C                                        SET UP MULTIPLE DATA COUNT.
0140        MULT=INT(AA(KN)+0.5)
0141        GO TO 16
0142  C.....................................................................
0143  C                                        END OF MULTIPLE DATA COUNT ?
0144     17 CONTINUE
0145        KM=KM+1
0146        IF(KM.GT.MULT) GO TO 15
0147  C.....................................................................
0148  C                                        NULL DATUM ?
0149        NCOUNT=NCOUNT+1
0150        IF(JDAT(KN).EQ.0) GO TO 13
0151  C.....................................................................
0152  C                                        STORE NUMBER IN APPROPRIATE
0153  C                                        ELEMENT OF C OR IC .
0154        LIM(II)=ICOUNT
0155        LIM(JJ)=JCOUNT
0156        LIM(KK)=KCOUNT
0157        IF(ININT.EQ.1) GO TO 18
0158        C(LIM(1),LIM(2),LIM(3))=AA(KN)
0159        GO TO 13
0160     18 CONTINUE
0161        IDUM=INT(AA(KN)+0.5)
0162        IF(AA(KN)_LE.-0.5) IDUM=IDUM-1
```

```
0163        IC(LIM(1),LIM(2),LIM(3))=IDUM
0164        GO TO 13
0165  C.....................................................................
0166  C                                     CHECK ENDING INDICATOR INEND
0167  C                                     AND JUMP TO APPROPRIATE LABEL.
0168     19 CONTINUE
0169        IF(INEND.GE.IDIM) GO TO 21
0170        IF(NUM.NE.0) GO TO 20
0171        IF(INEND.EQ.1) GO TO 12
0172        IF(INEND.EQ.2) GO TO 11
0173     20 CONTINUE
0174        IF(INEND.EQ.1.AND.KCOUNT.NE.NK1) GO TO 12
0175        IF(JCOUNT.EQ.NJ1.AND.KCOUNT.EQ.NK1) GO TO 14
0176        IF(INEND.EQ.2) GO TO 11
0177        GO TO 14
0178  C.....................................................................
0179  C                                     SET ERROR INDICATOR IERR .
0180     21 CONTINUE
0181        IERR=NCOUNT
0182        IF(INWAR.NE.0) IERR=-IERR
0183  C.....................................................................
0184  C                                     RETURN.
0185        RETURN
0186        END
**** LIST END ****
```

```
0001  C*****************************************************************
0002  C
0003  C          FORTRAN FREE-FORMAT DATA INPUT SCHEME
0004  C
0005  C          S : FFORM
0006  C
0007  C          G BUTLER & J PIKE : MAY '73
0008  C
0009  C*****************************************************************
0010  C
0011  C          FFORM READS A LINE OF CHARACTERS AND TRANSLATES THEM INTO
0012  C          NUMERICAL VALUES WHICH ARE STORED IN ARRAY AA.
0013  C
0014  C
0015  C          INPUT PARAMETERS
0016  C          ----------------
0017  C
0018  C          IDENT = IDENTIFIER & WARNING CONTROL
0019  C          ININT = 1 FOR READING INTEGERS, 0 OTHERWISE
0020  C          INWAR = WARNING COUNTER
0021  C          IUNIT = DATA INPUT UNIT
0022  C          NREC = DATA LINE COUNTER
0023  C
0024  C
0025  C          OUTPUT PARAMETERS
0026  C          -----------------
0027  C
0028  C          AA(40) = ARRAY FOR STORAGE OF NUMBERS FOUND
0029  C          INEND = ENDING INDICATOR
0030  C          INWAR = WARNING COUNTER
0031  C          JDAT(40) = ARRAY TO INDICATE IF CORRESPONDING AA VALUE
0032  C                     IS NULL OR MULTIPLE
0033  C          NUM = NUMBER OF VALUES FOUND
0034  C
0035  C
0036  C          INDICATORS
0037  C          ----------
0038  C
0039  C          INCOM - INDICATES THAT AN ALPHANUMERIC COMMENT HAS  STARTED
0040  C          INDAT - INDICATES WHETHER CURRENT VALUE IS NORMAL,
0041  C                  NULL OR MULTIPLE DATA
0042  C          INDIG - INDICATES THAT THE CURRENT CHARACTER IS A DIGIT
0043  C          INEXP - INDICATES AN EXPONENT HAS STARTED
0044  C          INIT - INDICATES THAT THE NUMERICAL ACCUMULATORS ARE
0045  C                 TO BE INITIALISED
0046  C          INMUL - INDICATES THAT A MULTIPLE VALUE HAS STARTED
0047  C          INNUM - INDICATES THAT A NUMBER HAS STARTED
0048  C          INPNT - INDICATES FIRST DECIMAL POINT FOUND
0049  C
0050  C
0051  C          CHARACTERS
0052  C          ----------
0053  C
0054  C          IBL = BLANK
0055  C          ICM = COMMA
0056  C          ID(10) = DIGITS 0 TO 9
0057  C          IDOL = DOLLAR
0058  C          IDO = DOUBLE QUOTES
0059  C          IEX = E
0060  C          IF = MINUS
```

```
0051   C          IPL = PLUS
0052   C          IPT = POINT
0053   C          ISL = SLASH
0054   C          IST = SINGLE QUOTES
0055   C          IBL = BLANK
0056   C
0057   C
0058   C          OTHER PARAMETERS
0059   C          ----------------
0060   C
0061   C          INP = CONTROL PARAMETER FOR WARNING RETURN
0062   C          J = CURRENT CHARACTER
0063   C          JCH(80) = CHARACTER STORAGE ARRAY
0064   C          JP = PREVIOUS CHARACTER
0065   C          JPOS(80) = WARNING POSITION STORAGE ARRAY
0066   C          JQ = CHARACTER BEFORE JP
0067   C          K = COUNTER FOR CHARACTERS
0068   C          KNUL = POSITION OF * FOR MULTIPLE DATA
0069   C          KP = LAST NON-BLANK CHARACTER
0070   C          KPOS = POSITION OF LAST NON-BLANK CHARACTER
0071   C          LE = ACCUMULATOR FOR EXPONENT
0072   C          LSG = SIGN OF EXPONENT
0073   C          LY = ACCUMULATOR FOR DECIMAL PLACES
0074   C          NCH = NUMBER OF CHARACTERS PER LINE
0075   C          NUM = COUNTER FOR NUMBERS FOUND
0076   C          SG = SIGN OF NUMBER
0077   C          X = ACCUMULATOR FOR DIGITAL VALUES
0078   C
0079   C
0080   C*************************************************************************
0081   C
0082   C
0083         SUBROUTINE FFORM(IUNIT,IDENT,NREC,INEND,INWAR,NUM,AA,JDAT,ININT)
0084         DIMENSION JCH(81),ID(10),JPOS(80),AA(40),JDAT(40)
0085         DATA IDI(1)/1H0/,IDI(2)/1H1/,IDI(3)/1H2/,IDI(4)/1H3/,
0086        1      IDI(5)/1H4/,IDI(6)/1H5/,IDI(7)/1H6/,IDI(9)/1H7/,
0087        2      IDI(9)/1H8/,IDI(10)/1H9/,
0088        3      ISL/1H /,ICM/1H,/,IPT/1H./,IMI/1H-/,IPL/1H+/,
0089        4     IEX/1HE/,ISQ/1H'/,IDQ/1H"/,IDOL/1H$/,IET/1H=/,ISL/1H//
0090   C.....................................................................
0091   C                                          READ LINE OF CHARACTERS
0092   C                                          & SET STARTING VALUES.
0093         TEN=10.0
0094         NCH=80
0095         DO 1 M=1,NCH
0096         JCH(M)=IBL
0097         JPOS(M)=0.
0098       1 CONTINUE
0099         NCHP1=NCH+1
0100         JCH(NCHP1)=IBL
0101         READ(IUNIT,500) (JCH(M),M=1,NCH)
0102         INEND=0
0103         INCOM=0
0104         INMUL=0
0105         INWAR=INWAR
0106         NUM=0
0107         J=ICH
0108         JP=J
0109         KP=J
0110         K=0
```

```
0121        INIT=1
0122  C.....
0123  C                                          LABEL 1000 .
0124   1009 CONTINUE
0125  C.....
0126  C                                          INITIALISE NEW NUMBER
0127  C                                          IF REQUIRED.
0128        IF(INIT.NE.1) GO TO 2
0129        SG=1.0
0130        K=0.0
0131        LY=0
0132        LZ=0
0133        LE=0
0134        LSG=1
0135        INNUM=0
0136        INPT=0
0137        INEXP=0
0138        INDAT=1
0139        INIT=0
0140     2 CONTINUE
0141  C.....
0142  C                                          SELECT NEXT CHARACTER.
0143        K=K+1
0144        JO=J
0145        J=ICHAR(K)
0146        IF(J.NE.IBL) KP=J
0147        IF(J.NE.IBL) KPOS=K-1
0148        J=JCHAR(J)
0149  C.....
0150  C                                          IS CHARACTER PART OF COMMENT ?
0151        IF(INCOM.EQ.1) GO TO 5000
0152  C.....
0153  C                                          HAS ENDING CHARACTER OCCURRED ?
0154        IF(INEND.GT.0) GO TO 5000
0155  C.....
0156  C                                          HAS EXPONENT STARTED ?
0157        IF(INEXP.GT.0) GO TO 3000
0158  C.....
0159  C                                          CHECK FOR NULL DATA AND PRINT
0160  C                                          WARNING IF MULTIPLE NULL DATA
0161  C                                          ERRORS DETECTED.
0162        II=0
0163        IF(J.EQ.ICM) II=1
0164        IF(J.EQ.ICL) II=1
0165        IF(J.EQ.INUL) II=1
0166        IF(II.EQ.0) GO TO 3
0167        IF(JO.EQ.ICM) INPUT=0
0168        IF(INNUM.GT.0) GO TO 3
0169        IF(INNUM.NE.1) GO TO 3
0170        IF(INPUT.EQ.KPOS) INDAT=0
0171        IF(INDAT.EQ.KPOS) GO TO 3
0172        IE=5
0173        JDATANUM=1
0174        CALL ERR
0175        IPRT=NULL
0176        GO TO 30
0177     3 CONTINUE
0178  C.....
0179  C                                          HAS NULL DATA BEEN FOUND ?
0180        IF(INDAT.EQ.0) GO TO 4000
```

```
0121  C.....
0122  C                                        CHECK FOR DIGIT.
0123        INDIG=0
0124        DO 4 M=1,10
0125        IF(J.EQ.IDI(M)) INDIG=1
0126     4 CONTINUE
0127  C.....
0128  C                                        HAS NUMBER STARTED ?
0129        IF(INNUM.GT.0) GO TO 2000
0130  C.....
0131  C                                        IS CHARACTER A DIGIT?
0132        IF(INDIG.EQ.0) GO TO 5000
0133  C.....
0134  C                                        START NUMBER.
0135        INNUM=1
0136  C.....
0137  C                                        CHECK WHETHER CHARACTER
0138  C                                        BEFORE NUMBER IS STANDARD.
0139        IF(JP.EQ.IPL) INNUM=2
0140        IF(JP.EQ.IMI) INNUM=2
0141        IF(JP.EQ.IMI) SG=-1.0
0142        IF(JP.EQ.IBL) INNUM=NCH
0143        IF(JP.EQ.ICM) INNUM=NCH
0144        IF(JP.EQ.IDQ) INNUM=NCH
0145        IF(JP.EQ.ISQ) INNUM=NCH
0146        IF(JP.NE.IST) GO TO 5
0147        INPNT=1
0148        LZ=1
0149        LY=-1
0150        IF(JQ.EQ.IPL) INNUM=3
0151        IF(JQ.EQ.IMI) INNUM=3
0152        IF(JQ.EQ.IMI) SG=-1.0
0153        IF(JQ.EQ.IBL) INNUM=NCH
0154        IF(JQ.EQ.ICM) INNUM=NCH
0155        IF(JQ.EQ.IDQ) INNUM=NCH
0156        IF(JQ.EQ.ISQ) INNUM=NCH
0157        IF(INMUL.EQ.1.AND.JQ.EQ.IST) INNUM=2
0158     5 CONTINUE
0159  C.....
0160  C                                        CHECK FOR MULTIPLE DATA.
0161  C                                        PRINT WARNING IF NON-COMPACT.
0162        IF(INMUL.NE.1) GO TO 6
0163        KPOS=K-INNUM
0164        IF(KMUL.EQ.KPOS) INNUM=2
0165        IF(KMUL.EQ.KPOS) GO TO 6
0166        LM=5
0167        JOPT(NUM)=1
0168        INMUL=0
0169        LEFT=KMUL
0170        GO TO 900
0171  C.....
0172  C                                        PRINT WARNING FOR NON-STANDARD
0173  C                                        CHARACTER BEFORE NUMBER.
0174     6 CONTINUE
0175        IF(INNUM.GT.1) GO TO 2000
0176        JW=1
0177        GO TO 900
0178  C.....
0179  C                                        LABEL 2002.
0180   2000 CONTINUE
```

```
0241  C.....................................................................
0242  C                                          BUILD UP NUMBER VALUE.
0243        LY=LY+LZ
0244        X1=X*10.0
0245        DO 7 M=1,10
0246        IF(J.EQ.IDI(M)) X=FLOAT(M-1)+X1
0247      7 CONTINUE
0248  C.....................................................................
0249  C                                          IS CHARACTER AN E?
0250  C                                          IF SO START EXPONENT.
0251        IF(J.EQ.IEX) INEXP=1
0252        IF(J.EQ.IEX) GO TO 3000
0253  C.....................................................................
0254  C                                          IS CHARACTER FIRST DECIMAL
0255  C                                          POINT OR DIGIT ?
0256        IF(J.EQ.IPT.AND.INPNT.NE.1) INDIG=2
0257        IF(J.EQ.IPT) INPNT=1
0258        IF(INDIG.EQ.0) GO TO 4000
0259  C.....................................................................
0260  C                                          IS CHARACTER AT END OF LINE ?
0261        IF(K.EQ.NCHP1) GO TO 4000
0262  C.....................................................................
0263  C                                          IF CHARACTER IS DEC. POINT,
0264  C                                          CHANGE NUMBER VALUE ACCUMULATORS.
0265        IF(J.EQ.IPT) LZ=1
0266        IF(J.EQ.IPT) LY=-1
0267        GO TO 5000
0268  C.....................................................................
0269  C                                          LABEL 3000 : UPDATE EXPONENT.
0270   3000 CONTINUE
0271  C.....................................................................
0272  C                                          IS THIS THE FIRST CHARACTER
0273  C                                          AFTER E ?
0274        IF(INEXP.GT.1) GO TO 8
0275  C.....................................................................
0276  C                                          IS CHARACTER BLANK + OR - ?
0277        IF(J.EQ.IPL) INEXP=2
0278        IF(J.EQ.IPL) INEXP=2
0279        IF(J.EQ.IMI) INEXP=2
0280        IF(J.EQ.IMI) LSG=-1
0281        IF(INEXP.GT.1) GO TO 5000
0282  C.....................................................................
0283  C                                          BUILD UP EXPONENT VALUE.
0284      8 CONTINUE
0285        LE=LE*10
0286        INDIG=0
0287        DO 9 M=1,10
0288        IF(J.NE.IDI(M)) GO TO 9
0289        LE=(M-1)+LE
0290        INEXP=3
0291        INDIG=1
0292      9 CONTINUE
0293  C.....................................................................
0294  C                                          IF NO DIGIT HAS OCCURRED IN
0295  C                                          EXPONENT, PRINT WARNING.
0296        IF(INEXP.EQ.3) GO TO 10
0297        IMP=2
0298        GO TO 300
0299     10 CONTINUE
0300  C.....................................................................
```

```
0631  C                                       IS CHARACTER A DIGIT ?
0632        IF(INDIG.EQ.0) GO TO 4000
0633  C.....................................................................
0634  C                                       IS CHARACTER AT END OF LINE ?
0635        IF(K.EQ.NCHP1) GO TO 4000
0636        GO TO 5000
0637  C.....................................................................
0638  C                                       LABEL 4000.
0639   4000 CONTINUE
0640  C.....................................................................
0641  C                                       CHECK WHETHER CHARACTER AT END
0642  C                                       OF NUMBER IS STANDARD.
0643        IF(J.EQ.IBL) GO TO 12
0644        IF(J.EQ.IPL) GO TO 12
0645        IF(J.EQ.IMI) GO TO 12
0646        IF(J.EQ.ICMA) GO TO 12
0647        IF(J.EQ.IDO) GO TO 12
0648        IF(J.EQ.IDS) GO TO 12
0649        IF(J.EQ.(-1)) GO TO 12
0650        IF(J.EQ.ISL) GOTO 12
0651        IF(J.NE.IST) GO TO 1
0652  C.....................................................................
0653  C                                       IF CHARACTER IS * PREPARE FOR
0654  C                                       MULTIPLE DATA.
0655        IF(INMUL.GE.1) GO TO 11
0656        IF(INPNT.GT.0) GO TO 11
0657        IF(INEXP.GT.0) GO TO 11
0658        IF(IG.LT.0.0) GO TO 11
0659        IF(X.LT.0.5) GO TO 11
0660        INMUL=2
0661        INDAT=2
0662        INUL=K
0663        GO TO 12
0664     11 CONTINUE
0665  C.....................................................................
0666  C                                       PRINT WARNING FOR NON-STANDARD
0667  C                                       CHARACTER AT END OF NUMBER.
0668        IF(K.EQ.NCH.AND.INDIG.EQ.1) GO TO 12
0669        IMP=1
0670        KFAT=K
0671        GO TO 200
0672     12 CONTINUE
0673  C.....................................................................
0674  C                                       STORE NUMBER VALUE IN ARRAY AA
0675  C                                       AND SET ARRAY IDAT TO SHOW NULL
0676  C                                       OR MULTIPLE DATA.
0677        NUM=NUM+1
0678        LP=LSGNLE-LY
0679        AA(NUM)=IG*XX*TEN**LP
0680        IDAT(NUM)=INDAT
0681        INUL(NUM)=INMUL-1
0682  C.....................................................................
0683  C                                       IF INTEGER READ ROUTINE, CHECK
0684  C                                       FOR INTEGER VALUE. IF NON-INTEGER
0685  C                                       FOUND, PRINT WARNING.
0686        IMP=1
0687        IF(INPNT.NE.1) GO TO 13
0688        IF(INPNT.GE.0) IMP=0
0689        IF(INEXP.NE.0) IMP=1
0690        IF(IMP.EQ.0) GO TO 13
```

```
0361        II=INT(AA(NUM)+0.5)
0362        IF(AA(NUM).LE.-0.5) II=II-1
0363        IMP=3
0364        GO TO 999
0365     13 CONTINUE
0366  C.....
0367  C                                        SET INITIALISATION FLAG
0368  C                                        FOR NEW NUMBER.
0369        INIT=1
0370  C.....
0371  C                                        LABEL 5000 .
0372   5000 CONTINUE
0373  C.....
0374  C                                        CHECK FOR COMMENT START OR END.
0375        IF(J.EQ.IDQ) INCOM=INCOM+1
0376        IF(J.EQ.IDD) INCOM=INCOM+1
0377        IF(INCOM.GT.1) INCOM=0
0378        IF(INCOM.EQ.1) GO TO 14
0379  C.....
0380  C
0381  C                                        CHECK FOR $ OR / AND SET
0382  C                                        ENDING INDICATOR.
0383        IF(J.EQ.ISL) INEND=4
0384        IF(INEND.EQ.4) GO TO 14
0385        IF(J.NE.IDOL) GO TO 14
0386        INEND=1
0387        IF(JP.NE.IDOL) GO TO 14
0388        INEND=2
0389        IF(JQ.NE.IDOL) GO TO 14
0390        INEND=3
0391     14 CONTINUE
0392  C.....
0393  C                                        IS CHARACTER AT END OF LINE ?
0394        IF(K.NE.NCHF1) GO TO 1000
0395  C.....
0396  C                                        IF MULTIPLE DATA DEFINITION
0397  C                                        UNFINISHED, PRINT WARNING.
0398        IF(INMUL.NE.1) GO TO 15
0399        IMP=7
0400        JDAT(NUM)=1
0401        KRAT=KMUL
0402        GO TO 999
0403     15 CONTINUE
0404  C..... .....
0405  C                                        PRINT FURTHER WARNING
0406  C                                        INFORMATION IF REQUIRED.
0407        IF(IDENT.EQ.0) GO TO 17
0408        IF(IDENT.GT.999) GO TO 16
0409        IF(INWAR.EQ.INWAS) GO TO 17
0410     16 CONTINUE
0411        M=NCHF1
0412     21 M=M-1
0413        IF(JCH(M).EQ.IBL.AND.M.NE.1) GO TO 21
0414        WRITE(59,540) (JCH(K),K=1,M)
0415        M=NCHF1
0416     22 M=M-1
0417        IF(JPOS(M).EQ.IBL.AND.M.NE.1) GO TO 22
0418        IF(INWAR.NE.INWAS) WRITE(59,540) (JPOS(K),K=1,M)
0419        IF(IDENT.NE.999) GO TO 17
0420        IF(NUM.NE.0) WRITE(59,550) (AA(K),K=1,NUM)
```

```
0420      47 CONTINUE
0421  C.....
0422  C                                        RETURN.
0424        RETURN
0425  C.....
0426  C                                        PRINT WARNINGS IF REQUIRED.
0427    999 CONTINUE
0428        IF(T.NE.37.0) WRITE(50,570)
0429        CHK=CHK+1
0430        IF(NERR.EQ.0) GO TO 10
0431        DO 9 J=1,NERR  WRITE(6,580) IDENT,NREC,K,JP,J
0432        IF(IMP.EQ.2) WRITE(6,590) IDENT,NREC,K,JP,J
0433        IF(IMP.EQ.3) WRITE(6,600) IDENT,NREC,K,JP,J,II
0434        IF(IMP.EQ.4) WRITE(6,610) IDENT,NREC,KPRT,JCH(KPRT-1),JCH(KPRT)
0435     10 CONTINUE
0436        IF(INT.LT.4) KPRT=
0437        JCH(KPRT)=IP
0438        GO TO (20,30,40,50,15),IMP
0439  C.....
0440  C                                        FORMATS.
0441    500 FORMAT(80A1)
0442    510 FORMAT(1H ,10HFOR IDENT=,I5,6H,AFTER,I4,8H LINES &,I3,
0443       1    7H CHAR. ,2A1,12H STARTS NUMBER  )
0444    520 FORMAT(1H ,10HFOR IDENT=,I5,6H,AFTER,I4,8H LINES &,I3,
0445       1    7H CHAR. ,2A1,12H IN EXPONENT  )
0446    530 FORMAT(1H ,10HFOR IDENT=,I5,6H,AFTER,I4,8H LINES &,I3,
0447       1    7H CHAR. ,2A1,12H ENDS NUMBER  )
0448    540 FORMAT(1H ,80A1)
0449    550 FORMAT(1H ,6E12.6)
0450    560 FORMAT(1H ,10HFOR IDENT=,I5,4H,F5.1,E15.7,
0451       1    14H FOUND, INTERP,I7,8H NEEDED  )
0452    570 FORMAT(1H  72H ***** 1 OR MORE REAL WARNINGS *****  )
0453        END
***** LIST END *****
```

Appendix C

USERS' GUIDE TO FORTRAN FREE-FORMAT INPUT SCHEME

The scheme enables free-format data to be read into various arrangements of the prescribed variables. Each routine starts on a new line of data and continues until either the specified number of numbers has been read or an end of read symbol (/ or $) is found. The routines are designed to read anything. When ambiguous or unconventional data is encountered, the most reasonable interpretation is taken and a warning message is output. A read failure can only occur from a system failure, for example if the number is too big for the computer or an attempt is made to read beyond the end of a file.

Various calls are permitted to give maximum flexibility in the choice of input variables. The scheme is programmed in standard Fortran IV and has been tested on a wide range of computers. Data prepared in accordance with the latest FOR77 recommendations for free-format data input are compatible with the read routines.

The read routine calls

For real numbers,

n real numbers (where n represents an integer $1 \leqslant n \leqslant 9$)	READn
one-dimensional real array	READA
two-dimensional real array	READB
three-dimensional real array	READC

and for integer numbers, similarly,

n integer numbers	READIn
one-dimensional integer array	READIA
two-dimensional integer array	READIB
three-dimensional integer array	READIC .

The parameters are,

```
READn (IUNIT, IDENT, IERR, X1,...,Xn)
READA (IUNIT, IDENT, IERR, A, I1, I2)
READB (IUNIT, IDENT, IERR, B, I1, I2, NI, J1, J2, NJ, IORD)
READC (IUNIT, IDENT, IERR, C, I1, I2, NI, J1, J2, NJ, K1, K2,
       NK, IORD)
```

with those for the integer reads being identical, except that the real names X1,...Xn, A, B, C are replaced by the integer variables L1,...Ln. IA, IB and IC. In detail the parameters have the values:

IUNIT — input device number.

IDENT — identification number. This is a number which appears in warning messages to identify which read call has generated the warning. It is also used to control the type of warning message output. IDENT = 0 suppresses all error output and IDENT $\geqslant$ 1000 provides a complete list of the symbols read by the read routine.

IERR — is a variable name which is given a value at the end of the read call. Its value is set to sign *N where N is the number of numbers found and 'sign' is +1, except when a warning connected with ambiguous or unconventional data is given when it is set to -1. It should be noted that suppressing the output with IDENT = 0 does not affect the operation of IERR. It is anticipated that the main use of IERR will be to control 'batch' running (*eg* to prevent programs running if ambiguities are present in the initial data input).

X1,...,Xn — n real variable names.

A, I1, I2 — A is a one-dimensional real array name and the numbers are read into elements A(I1) to A(I2). That is (I2 - I1 + 1) numbers are read into consecutive locations in the array.

B, I1, I2, NI, J1, J2, NJ — B is a two-dimensional array of dimensions NI, NJ and the numbers are read into B(I,J) where I1 $\leqslant$ I $\leqslant$ I2 and J1 $\leqslant$ J $\leqslant$ J2 (see IORD).

C, I1, I2, NI, J1, J2, NJ, K1, K2, NK — C is a three-dimensional array of dimensions NI, NJ, NK, and the numbers are read into C(I,J,K) where I1 $\leqslant$ I $\leqslant$ I2, J1 $\leqslant$ J $\leqslant$ J2 and K1 $\leqslant$ K $\leqslant$ K2 (see IORD).

IORD — controls the order in which the arrays are filled. For two-dimensional arrays, if IORD = 21, the subscript I varies whilst J remains fixed (that is, columns are read in matrix terms). For all other values of IORD, J varies whilst I remains fixed (rows are read in matrix terms).

For three-dimensional arrays a similar system is used; for example IORD = 132 means that J (identified with 2) varies most rapidly, followed by K (identified with 3), followed by I (identified with 1). The other significant values of IORD are as follows:

IORD = 231 implies JKI order
213 implies JIK order
312 implies KIJ order
321 implies KJI order

For all other values of IORD, the order is assumed to be IJK.

Form of free format numbers

The read routines will read numbers in any of the following forms:

$\bar{S}d$, $\bar{S}.d$, $\bar{S}d.d$, $\bar{S}d.$ $\bar{S}d\bar{E}d$, $\bar{S}.d\bar{E}d$, $\bar{S}d.d\bar{E}d$, $\bar{S}.d\bar{E}d$

where d is a single digit or a compact string of digits, $\bar{S}$ is +, -, or no character, and $\bar{E}$ is an E, E blank, E+ or E-. Numbers are normally separated by blank or comma (delimiters). Other forms of number of delimiter will be read and accepted, but a warning message is given so that the user can check that the interpretation taken by the read routine is that intended. Some special characters may also be used without giving warning messages as is detailed below.

The following additional features have been introduced for convenience in using the system.

(1) Non-numeric comments can be inserted in the data, but a warning will be given if there is no delimiter between comments and numbers, *eg*

1.23, ABC, 2.45 will give no warning
but 1.23ABC2.45 will give two warnings.

(2) Alphanumeric comments may be inserted by enclosing them between inverted commas (either single or double). In this case, the inverted comma is allowed as a delimiter, *eg*

1.23"AB3"2.45 will give no warning.

(3) Null data (*ie* variables retain their previous values) are denoted by commas optionally enclosing blanks, *eg* 123,, , 345 gives four numbers: 123, two null data and 345. *Note* that comma occurring as the first non-blank character on a line implies a null datum, but that comma occurring as the last non-blank character on a line does not imply an extra null-datum, *eg* , 123, 345, gives three numbers: 1 null datum, 123 and 345.

(4) Multiple data are denoted by i* number where i is a non-negative integer, *eg*

3*3.14159 means three values of 3.14159

3* , means three null data.

When * is used in this way, no warning is generated.

(5) The character / ends the read call forthwith, and any further characters beyond the slash on the same line are ignored. Thus / can be used for the protection of further data or as a method of retaining the existing values for all further variables of the read call.

For all read calls except the two-dimensional and three-dimensional array reads, $ has the same effect as /. For two-dimensional array reads $ stops the current one-dimensional sub-array and $$ stops the read. For three-dimensional array reads $ stops the current one-dimensional sub-array, $$ stops the current two-dimensional sub-array and $$$ ends the read. *Note* that $ and / will have no effect if included between inverted commas.

(6) Integer locations are set to the nearest whole number when supplied with real data and a warning is given. It should be noted that integers and reals cannot be mixed in the same read. Hence a Fortran

```
READ (IUNIT, FORMAT)K, A(K), B(K), C(K)
```

would need to be replaced by

```
CALL READ4 (IUNIT, IDENT, IERR, X1, X2, X3, X4)
K = INT (X1 + 0.5)
A(K) = X2
B(K) = X3
C(K) = X4 .
```

Example:

To read three numbers into the array PTS at locations 4 and 6 from input device 5, use

```
CALL READA (5, 1, IERR, PTS, 4, 6) .
```

If the characters read from device 5 are

```
X = 3.14    Y = .3E1    Z =    2
```

then a warning is given because the number 3.14 starts with an = symbol (only space or , are permitted). The form of the warning is

```
***** DATA READ WARNINGS *****

FOR IDENT = 1, AFTER 1 LINE AND 3 CHAR.  =3 STARTS NUMBER
X = 3.14     Y = .3E1     Z = 2
    +
```

PTS (4) to PTS (6) are set to the values 3.14, 3 and 2 respectively.

REPORT DOCUMENTATION PAGE

Overall security classification of this page

UNLIMITED

As far as possible this page should contain only unclassified information. If it is necessary to enter classified information, the box above must be marked to indicate the classification, e.g. Restricted, Confidential or Secret.

1. DRIC Reference (to be added by DRIC)	2. Originator's Reference	3. Agency Reference	4. Report Security Classification/Marking
	RAE TM Aero 1804	N/A	UNLIMITED

5. DRIC Code for Originator	6. Originator (Corporate Author) Name and Location
7673000W	Royal Aircraft Establishment, Farnborough, Hants, UK
5a. Sponsoring Agency's Code	6a. Sponsoring Agency (Contract Authority) Name and Location
N/A	N/A

7. Title A free-format data input scheme written in Fortran IV

7a. (For Translations) Title in Foreign Language

7b. (For Conference Papers) Title, Place and Date of Conference

8. Author 1. Surname, Initials	9a. Author 2	9b. Authors 3, 4	10. Date	Pages	Refs.
Butler, G.F.	Pike, J.	-	May 1979	41	-

11. Contract Number	12. Period	13. Project	14. Other Reference Nos.
N/A	N/A		

15. Distribution statement
(a) Controlled by –
(b) Special limitations (if any) –

16. Descriptors (Keywords) (Descriptors marked * are selected from TEST)

Computer programming*. Fortran*.

17. Abstract

Data for Fortran IV programs has to be in fixed format. Some users find this constraint irksome. Presented here is a data input scheme for Fortran IV which makes the minimum of demands on the data structure. If the data is well-formed and unambiguous it will be read. Ill-formed and ambiguous data will also be read and given a reasonable interpretation, with the location of the suspect data and the value assumed being output as a (suppressible) error message. The scheme also allows input variables to retain their previous values, identical consecutive data to be input in a simplified form and alphanumeric comments to appear amongst the data. The whole data input scheme is written in standard Fortran IV so that the advantages of machine transferability of the program are retained.